2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

An ASABE Meeting Presentation

Paper Number: 096138

Three-dimensional Reconstruction of Fruit Trees by a Shape from Silhouette Method

Amy Tabb

Agricultural Engineer, Appalachian Fruit Research Station, Agricultural Research Service, USDA

Graduate Student, Robot Vision Lab, Electrical and Computer Engineering Department, Purdue University

Written for presentation at the
2009 ASABE Annual International Meeting
Sponsored by ASABE
Grand Sierra Resort and Casino
Reno, Nevada
June 21 – June 24, 2009

Abstract. *In order to robotically prune a dormant fruit tree, the branches must be identified in three-dimensional space. Furthermore, the branches need to be measured in order to determine which branches should be pruned. Both the identification and measurement of branches can be accomplished by generating a three-dimensional reconstruction of the tree, and by making measurements on the reconstruction. In this work, we use a computer vision-based method to generate the three-dimensional reconstruction. Images of the tree are acquired in a laboratory setting, and then a shape from silhouette (SFS) method of our own design is used to reconstruct the shape of the object. Although we use the SFS method for leafless apple trees, this method is appropriate for many complex objects. We present an overview of our method of SFS and experimental results in a laboratory setting.*

Keywords. Three-dimensional reconstruction, robotic, pruning, apple, modeling, computer vision

The authors are solely responsible for the content of this technical presentation. The technical presentation does not necessarily reflect the official position of the American Society of Agricultural and Biological Engineers (ASABE), and its printing and distribution does not constitute an endorsement of views which may be expressed. Technical presentations are not subject to the formal peer review process by ASABE editorial committees; therefore, they are not to be presented as refereed publications. Citation of this work should state that it is from an ASABE meeting paper. EXAMPLE: Author's Last Name, Initials. 2009. Title of Presentation. ASABE Paper No. 09----. St. Joseph, Mich.: ASABE. For information about securing permission to reprint or reproduce a technical presentation, please contact ASABE at rutter@asabe.org or 269-429-0300 (2950 Niles Road, St. Joseph, MI 49085-9659 USA).

Mention of trade names or commercial products in this publication is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the U.S. Department of Agriculture.

The authors are solely responsible for the content of this technical presentation. The technical presentation does not necessarily reflect the official position of the American Society of Agricultural and Biological Engineers (ASABE), and its printing and distribution does not constitute an endorsement of views which may be expressed. Technical presentations are not subject to the formal peer review process by ASABE editorial committees; therefore, they are not to be presented as refereed publications. Citation of this work should state that it is from an ASABE meeting paper. EXAMPLE: Author's Last Name, Initials. 2009. Title of Presentation. ASABE Paper No. 09----. St. Joseph, Mich.: ASABE. For information about securing permission to reprint or reproduce a technical presentation, please contact ASABE at rutter@asabe.org or 269-429-0300 (2950 Niles Road, St. Joseph, MI 49085-9659 USA).

1 Introduction

Pruning dormant fruit trees is a necessary task for the health of trees and quality of fruit. By dormant pruning, we mean pruning that is done while the tree in its dormancy phase, during the winter. As a result, the leaves are not present. Dormant pruning requires a highly skilled workforce during one of the most uncomfortable times of year to be outside, the winter. Currently, automatic cutters are available. Even so, workers must repeatedly perform the same motions with their arms over their heads, again and again, in order to accomplish pruning. The popularity of high-density training systems for many specialty crops, particularly apple, have led to uniform orchards with a sparse branching structure. If the arrangement and diameters of branches could be determined then with some assistance from horticulturalists it may be possible to design a decision system for pruning.

For instance, in apple there is currently a simple rule called the fifty percent rule. Trees have a central leader branch, and then secondary branches emanating from the central leader. During pruning, the secondary branches are removed at the point where they join the central leader. The rule states that secondary branches whose diameter is larger than fifty percent than the diameter of the central leader branch are pruned.

In order to get the information needed for the fifty percent rule, or any other rule, for that matter, a three-dimensional reconstruction of the tree is needed. By three-dimensional reconstruction, we mean that data about the tree is acquired by sensors, and then this data is merged in some way to generate the position and shape of the tree in three-dimensional space, $\mathbb{R}^3$. A three-dimensional reconstruction is needed for the robotic vision system, not only to make decisions about what needs to be done to the object, but also to avoid colliding with it. In this article, we present some preliminary steps towards robotic pruning by presenting a method for the three-dimensional reconstruction of fruit trees in the laboratory. Even though pruning is our eventual goal, the method is general and can be applied to a range of objects and applications.

Our preliminary vision system for the reconstruction of complex objects such as dormant apple trees uses a computer vision method called shape from silhouette (SFS). Images are acquired of the object, and the regions in images corresponding to the target object are segmented from non-object regions, generating silhouettes. For each image, the silhouette is backprojected into $\mathbb{R}^3$. The intersection of all of the backprojected silhouettes produces an approximate reconstruction of the object called the *visual hull*. The visual hull will be discussed in more detail in sections 3 and 4, but we mention now that for objects that satisfy certain shape constraints, the visual hull can be an accurate reconstruction method.

The organization of this article is as follows. First we present some characteristics of leafless apple trees, and explain why the classical methods for three-dimensional reconstruction fail, in section 2. We present the recent literature concerning SFS in section 3. We explain the methodology of our shape from silhouette method, including pertinent definitions as well as constraints that must be satisfied by the object and cameras in section 4. Section 5 shows the experimental results of our algorithm when applied to leafless apple trees in a laboratory environment. Finally, in section 6 we present a conclusion about the work and also future research directions.

2 Characteristics of leafless apple trees and their images

In this section, we explain some of the characteristics of the shape and images of leafless apple trees. See Fig. 1 for some images of the trees used in these experiments.

First of all, the shape of the tree is complex; in images of the tree, there are numerous self crossings. If we look at a tree and shift our viewpoint slightly, the image of the tree changes considerably, unlike many objects where close views are similar. Secondly, the local appearance of one section of a branch is not significantly different from the local appearance of a section of a different branch. The shape of the tree is complex and is characterized by convex and saddle-shaped regions, but few to no concavities.

The classical approach of three-dimensional construction as presented in (Hartley and Zisserman 2004) is that feature points are first extracted from the images. Then, correspondences of feature points are computed between images; corresponding points are triangulated to form a point cloud in $\mathbb{R}^3$. The points can then be meshed or subjected to further processing.

When an object has the characteristics we mentioned here, the classical approach fails. First of all, established methods for feature point detection such as the Harris corner operator or even newer methods such as SIFT (Lowe 1999) discover feature points on the boundary of the object and the background. When the view changes slightly, frequently there are no feature points corresponding to the same three-dimensional position as the feature points in the first image. Clearly, we must look for another approach.

While the uniformity of the tree surfaces is a detrimental characteristic if we use the classical approach, this characteristic is actually an asset if we consider using the silhouettes of the object to reconstruct the shape of the tree. A background subtraction method, or some other method, extracts the object regions from the background, and then a shape from silhouette method reconstructs the three-dimensional shape. In the next section, we present a survey of recent literature in this field, called shape from silhouette, or SFS.

3 Literature Review

In this section we will present a survey of recent literature on the reconstruction of complex objects using shape from silhouette. First we give some definitions that are frequently used in the shape from silhouette literature in 3.1. Theoretical work on SFS is presented in 3.2. Then we consider the two competing approaches for SFS, volumetric and surfaces approaches, in 3.3 and 3.4, respectively.

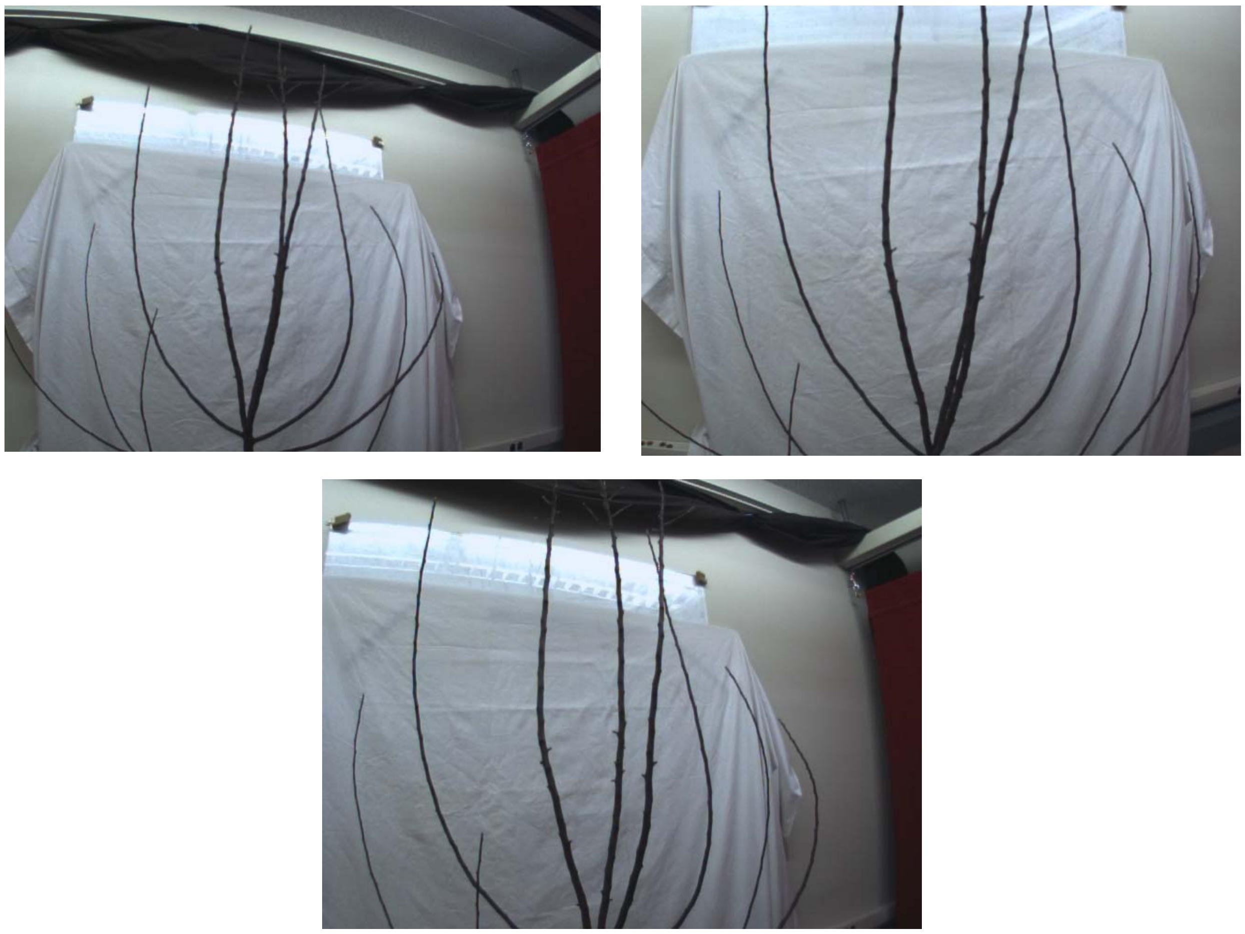


Figure 1. Images of the leafless apple tree used to compute the visual hull.

3.1 Definitions

Some terminology common to many shape from silhouette methods will be briefly presented here.

Consider an object **M** in $\mathbb{R}^3$, such as the temple figurine in Fig. 2.[1] If a camera is pointing at **M**, the *silhouette* is the projection of **M** to the image plane. A standard practice in SFS is that the pixels corresponding to the silhouette are labeled white as in Fig. 3. If we backproject the pixels in the silhouette (residing in the image plane), a set of rays into $\mathbb{R}^3$ called the *viewing cone* is generated. Note that every ray in the viewing cone intersects or is tangent to **M**.

The rays in the viewing cone that are tangent to **M** touch **M** at a set of points; this set is called the *contour generator*, or *rim*. Projecting the rim to the image plane generates the *occluding contour*, as in Fig. 4. The relationship between the rims and occluding contour is also illustrated in Fig. 5.

[1] Images of the temple dataset throughout this document are used with permission from the Middlebury College multi-view dataset, (Seitz et al 2006).

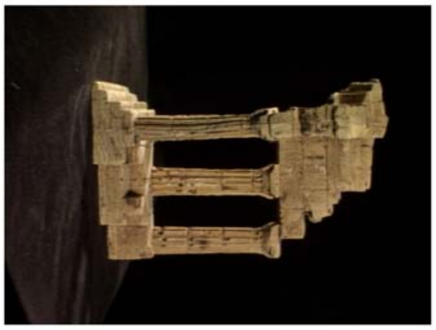


Figure 2. Input image of a temple figurine.

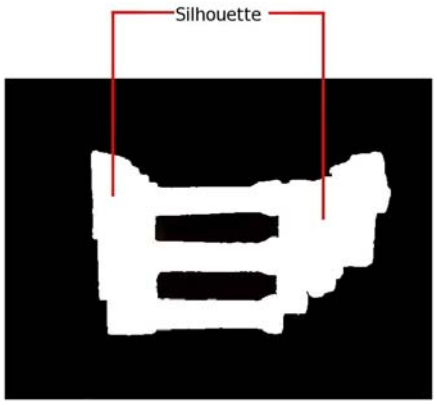


Figure 3. The silhouette of the input image consists of the white regions.

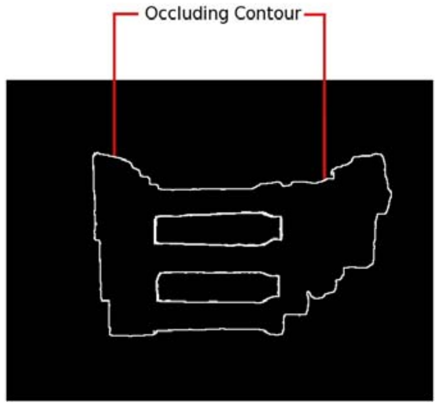


Figure 4. The occluding contour is the boundary of the silhouette and non-silhouette regions; shown in white.

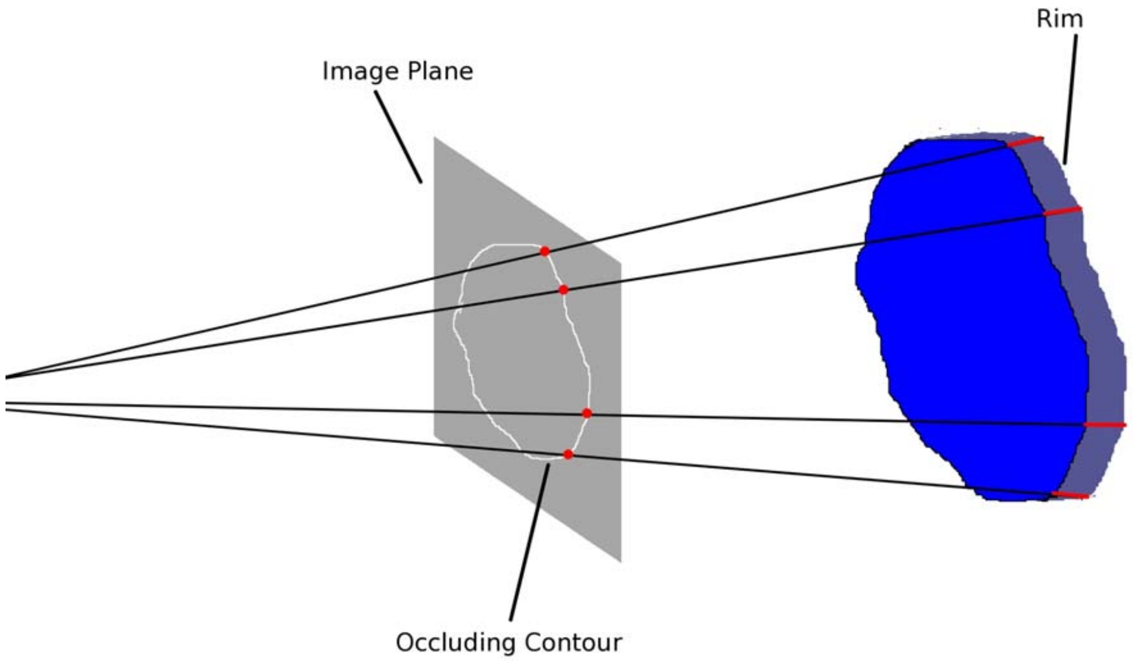


Figure 5. An illustration of the relationship between the occluding contour and rims. The image plane is indicated in the figure; the occluding contour is indicated in white on the image plane. Some viewing cone rays are shown; they touch the occluding contour and the target object (in blue) at the points labeled in red. The rim is the shaded part of the target object, and represents all regions where the viewing cone rays are tangent to the object.

3.2 The theory of shape from silhouette

Koenderink (1984) and Richards et al (1987) characterized the shape that can be inferred from silhouettes. They both use an interpretation rule whereby the maximal three-dimensional shape that could have generated the silhouettes is inferred. Also, Koenderink and van Doorn (1976) show that when the interpretation rule is used, saddle-shaped and locally cylindrical three-dimensional surfaces can be inferred from convex and saddle-shaped occluding contours, respectively. Given a convex occluding contour, convex or concave surfaces could be inferred. This ambiguity is resolved by the interpretation rule, which specifies that the maximal possible surface is inferred. Consequently, if the occluding contour is convex, we infer a convex surface. The implication of these results is that concave three-dimensional surfaces are never inferred.

Laurentini (1994) discussed the properties of the three-dimensional shape that is inferred given an infinite number of silhouettes. This shape, the visual hull, has several properties that are relevant to SFS. The first is that the true object is guaranteed to lie within the visual hull; if we combine this property with the results given in the previous paragraph, we can see that given an infinite number of cameras and an object with no concavities, the visual hull is a very accurate reconstruction of the object. The second property is that the visual hull is *silhouette consisten*t. What this means is that when the shape is projected to every image, the projected shape exactly matches the silhouettes. While the visual hull ***VH*** is technically defined for an infinite number of cameras, Slabaugh et al. (2001) note that for a finite number of cameras the maximal silhouette consistent shape is the *inferred visual hull*.

The distinction between the visual hull, which is the maximal silhouette-consistent shape in the limit of an infinite number of cameras, and the inferred visual hull, which is the maximal silhouette-consistent shape for a finite set of cameras, is rarely made explicit in the literature. In this work we use the term visual hull for both entities, and only make a distinction when referring to the visual hull for an infinite number of cameras.

From a practical standpoint, the visual hull can be computed by backprojecting silhouettes in $\mathbb{R}^3$, and then intersecting the backprojected silhouettes from all cameras. The next two sections will explain two different approaches for performing the backprojection.

3.2 Volumetric approaches

The first attempts to solve the SFS problem were volumetric methods. Volumetric methods divide up the region of the object into discrete units called voxels. The voxels are labeled either on or off, representing whether a voxel is part of the reconstruction of the object or not. The shape of the voxels, the method of division, and the method of labeling differ by method. More detail concerning volumetric methods can be found in the surveys of the subject, (Dyer, 2001) and (Slabaugh et al., 2001), and in milestones in the field such as (Martin and Aggarwal, 1983, Chien and Aggarwal, 1986, Potmesil, 1987, Szeliski, 1993).

We mention here that if the object is very thin, such as in the case of leafless apple trees, volumetric methods get bogged down with the division of voxels into smaller and smaller units along the boundary of the object. In addition, the resolution of the reconstruction is determined by how small voxels are divided. Since our object is very thin, we do not consider volumetric methods, instead concentrating on surface approaches.

3.3 Surface-based approaches

Surface-based approaches typically exploit the properties associated with the occluding contour in order to characterize the surface of the visual hull.

Early works sought to approximate the depth of points on the surface and to approximate surface curvature through either differential geometry or epipolar constraints, such as in the following works of (Cipolla and Blake, 1990, Cipolla and Blake, 1992, Vaillant and Faugeras, 1992, Szeliski and Weiss, 1993, Szeliski and Weiss, 1998, Boyer and Berger, 1997). The camera positions must be very close together in order to accurately reconstruct the surface, and a key assumption is that the object be smooth. The occluding contours were approximated by continuous (smooth) functions. These early works were not able to reconstruct the entire surface, but instead generated approximations of the surface at some local points.

The next series of works dealt with the computation of the *exact visual hull*, which Lazebnik et al (2007) defined as the visual hull in terms of its intrinsic properties: the vertices, edges, and faces of the visual hull for a given set of cameras. All of these intrinsic properties can be determined via an intersection of the viewing cones from all cameras, a three-dimensional problem. If the method performs the intersection in $\mathbb{R}^3$, the method is said to be *world-based*, while if it performs the intersection by exploiting epipolar constraints between images, it is *image-based*. Another important aspect of the exact visual hull is that occluding contours were interpreted as piecewise linear polygons, obviating the need for approximation of contours as continuous functions.

Matusik et al (2000, 2001) used an image-based algorithm to perform the intersections of viewing cones, and generated a boundary representation of the visual hull. The authors used epipolar constraints on occluding contours and an efficient edge-binning method.

Others represented the exact visual hull with a polyhedral graph (Boyer and Franco, 2003, Lazebnik et al., 2001, Lazebnik and Ponce, 2005, Lazebnik et al., 2007, Franco and Boyer, 2003, Franco and Boyer, 2009, Franco 2005). These works sought to determine the vertices, edges, and faces of the graph. The visual hull polyhedron then can be displayed by walking half-edges in a specific orientation until all edges have been walked twice (once in each direction).

Of those using the polyhedral graph representation, Boyer and Franco (2003) produced a hybrid exact visual hull method. The first stage computed polyhedral graph vertices, while the second stage approximated the edges (and therefore, the faces of the visual hull) by performing a Delaunay triangulation of the vertices. In other words, the vertices were exact but edges and faces were approximated.

Lazebnik et al in (Lazebnik et al., 2001, Lazebnik and Ponce, 2005, Lazebnik et al., 2007) used an image-based method based on projective oriented geometry in order to determine the visual hull of smooth objects. The cameras need only be weakly calibrated, and the method uses incremental computation. However, there are some problems related to numerical robustness; specifically, the localization of an *intersection point*, which is a vertex of the graph where three or more viewing cone patches intersect in $\mathbb{R}^3$.

The work of Franco and Boyer in (Franco and Boyer 2003 and 2009, Franco 2005) on their method 'Exact Polyhedral Visual Hull,' or EPVH, is the first surface-based method to allow target objects to lie outside of the field of view of some cameras. EPVH is an image-based, batch method that uses epipolar constraints in order to compute the exact visual hull. While very efficient, a problem is the generation of some polyhedral graph edges called `hanging edges,' where one endpoint and the direction of the edge is known, but the other endpoint is unknown. A search is initiated from the known endpoint in the direction of the unknown one, but occasionally this search process can fail because of numerical stability issues, as shown in Liu et at (2007).

The only world-based method is that of Baumgart (1974). The occluding contours were approximated by polygons, and the projection of the polygons were intersected, forming the visual hull. Using the terminology of the introduction, this method is characterized as world-based, but the result is approximate because of the occluding contour approximation.

4 A new method for shape from silhouette

We desire that a SFS method for our application have the following characteristics. The first is that the method allow incremental, divide and conquer, or parallel implementation. The styles of implementation listed here allows the visual hull to be computed with a subset of cameras; then, more cameras can be added as needed to refine the result. One of the drawbacks of the image-based method is that there is no mechanism whereby the visual hull computed with one subset of cameras and the visual hull computed with another subset of cameras can be merged to produce the visual hull for all cameras. The second desired characteristic is that the method allow the object to be at least partially outside of the field of view of some cameras. Most SFS methods assume that the object is within the field of view of all cameras, but since the tree is such a large and complex object, it is not possible to capture the level of detail required about branching points and contain the tree in every image. We consider Franco's EPVH method (Franco and Boyer, 2009) to be one of the most promising methods from the literature review. However, EPVH allowed the second characteristic, but did not allow the first. Consequently, we design our own method for SFS by intersecting backprojected silhouettes in $\mathbb{R}^3$.

Recall that in section 2 we observed that the tree contains few to no concavities. In section 3.2, we discussed how SFS more closely approximates the true shape of the object if that object has no concavities. As a result, we propose SFS as a promising approach for reconstructing the three-dimensional shape of the trees. In this section, we give an overview of our SFS method and its application to the reconstruction of leafless apple trees.

First, we make explicit the assumptions under which we are working in section 4.1. Then, we show how to undistort images for SFS in section 4.2, and convert the occluding contours to planar facets in 4.3. Finally, we discuss the intersection of backprojected silhouettes in section 4.4.

4.1 Assumptions

Most works on SFS assume that the target object is within the field of view of all cameras; we call this the *total visibility assumption*. However, in our application, the total visibility assumption is not valid, because the tree is quite large and complex. Consequently, the object is only partially visible by each camera.

In contrast to the total visibility assumption for the tree reconstruction application, we assume that any region ***S*** which is unseen by any camera is occupied. First of all, this assumption allows for those situations when the total visibility assumption is not valid, as in our application. Since we eventually are considering a robot application, another advantage of our assumption is that it is safer to assume that regions are occupied if there is no evidence for the contrary. Otherwise, if we had assumed that a region ***S*** was unoccupied by following the total visibility assumption, then the robot arm could be moved through such a region, potentially damaging cameras, robot, and the tree. Our method can be adapted to situations where the total visibility assumption is used or not used; we alert the reader in the text to the necessary adjustments to the method depending on the situation.

We also assume that the camera calibration parameters, internal (including radial distortion) and external, are known for every camera. In addition, we assume that an accurate segmentation of the object from the background in the image is available. Fig. 6 shows a raw image and then its segmented version.

4.2 Radial distortion correction

As mentioned previously, we assume that the radial distortion parameters are known. We do not explain undistortion in detail in this section, but instead explain how to make adjustments to the undistortion process such that the visibility assumption is incorporated and information on the corners of the image is not lost. In order to see how our method is different, we must first mention the classical method of undistortion. Usually, the image is undistorted into an image of the same size. See Fig. 7 to see how the information in the corners of the image is lost. We wish to undistort the image such that we preserve the silhouette information in all regions of the image.

An overview of the process is as follows. First, we offset the original image by some amount of pixels into a larger image. If the total visibility assumption is in effect, we fill the region not occupied by the original image with the color black; if not, we fill with the color white. We transform the internal camera calibration matrix K to a new matrix K'. Then, we use the radial distortion parameters and the internal camera calibration matrix K' to undistort the larger image. This process is demonstrated in Fig. 8. We will now discuss each component of the process.

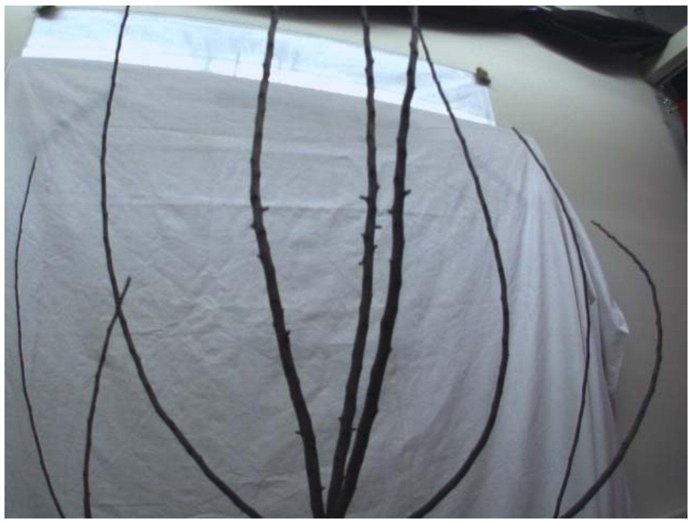
Let the number of offset (in pixels) in the x and y direction be called off_x and off_y; in our experiments, we set both offsets to 100 pixels. The original image's upper left hand corner is placed at location (off_x, off_y) in the larger image. As mentioned previously, we fill the border with black if the total visibility assumption is in effect, and fill the border with white if not. We then need to adjust the internal camera calibration matrix to account for the offset. We use the terminology of Hartley and Zisserman (2009), chapter 6: Camera Models, to represent the internal camera calibration matrix K of the original image:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where α_x and α_y are the focal lengths of the camera in terms of pixel dimensions for the two axes. The term s is the skew parameter. The terms x_0 and y_0 are the coordinates of the principal point in pixels. We adjust K in order to produce a new matrix, K', for the larger image:

$$K' = \begin{bmatrix} \alpha_x & s & x_0 + off_x \\ 0 & \alpha_y & y_0 + off_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

After K' has been computed, we undistort the larger image with the original radial distortion parameters and the internal calibration matrix K'.


(a) An image of the apple tree

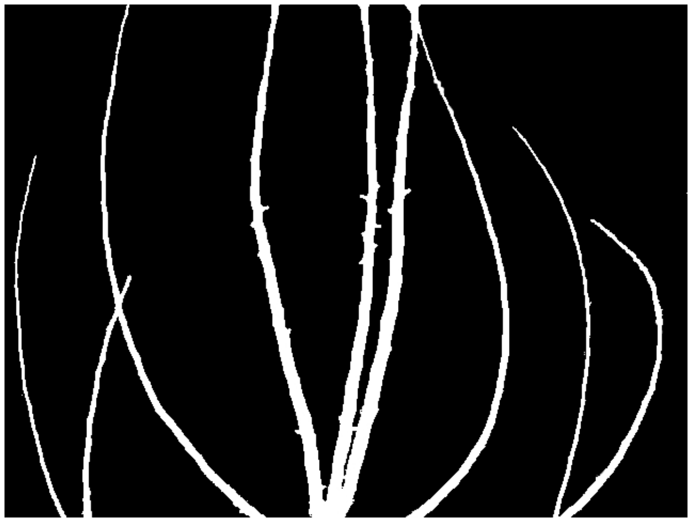

(b) The segmented image.

Figure 6. An example of the original image and the segmented image.

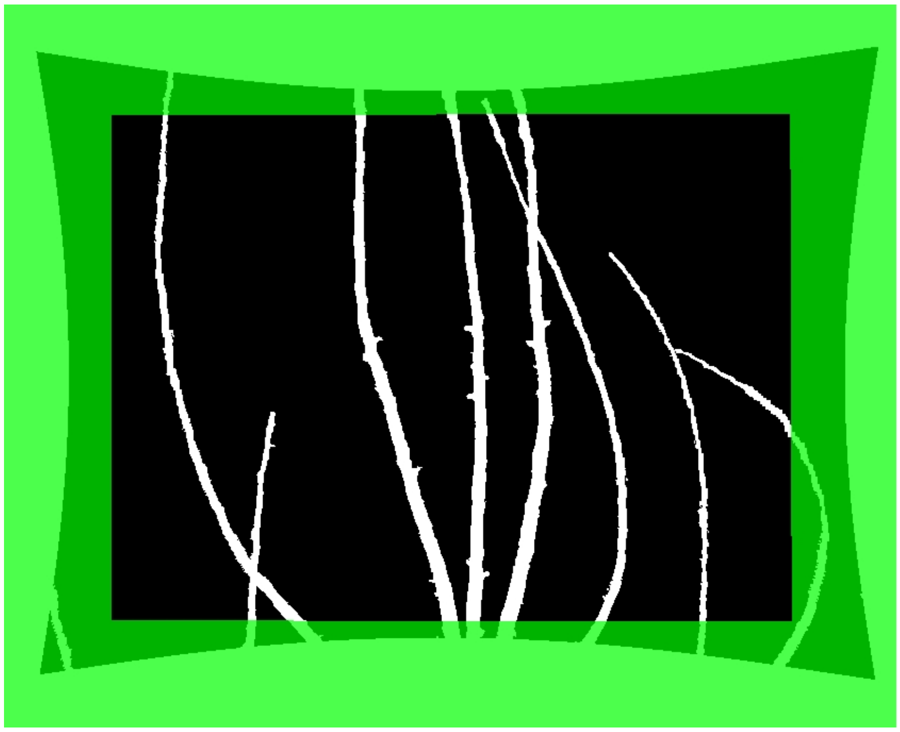


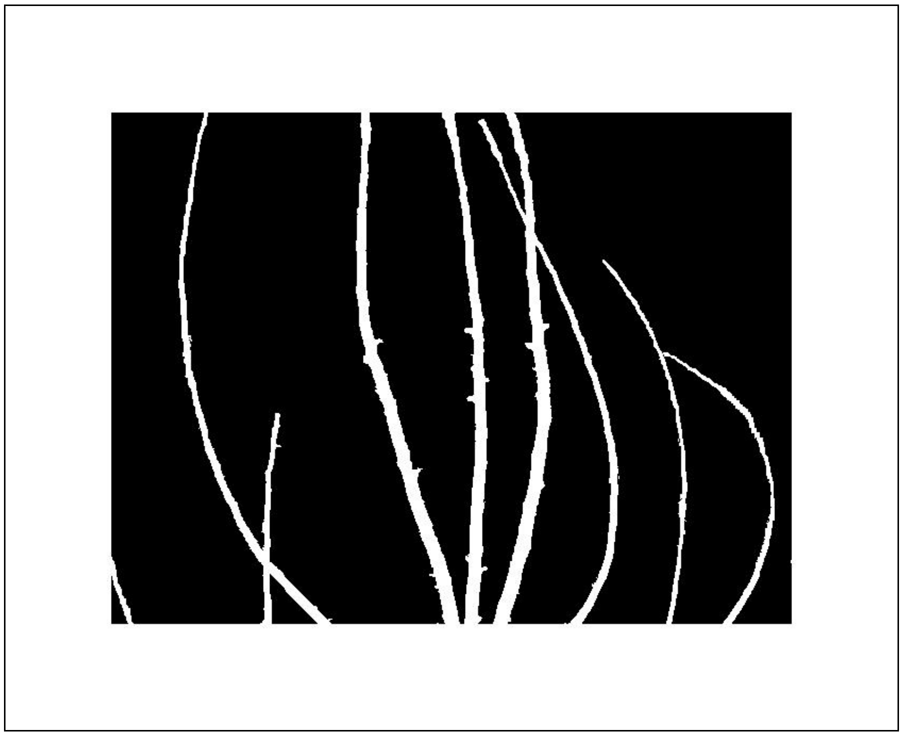
Figure 7. The region that is not overlaid with green is the result of using the classical undistortion method. Notice that the regions on the edges and corners, which are overlaid with green, would have been lost.

4. 3 Contour extraction and conversion of contours to planar facets

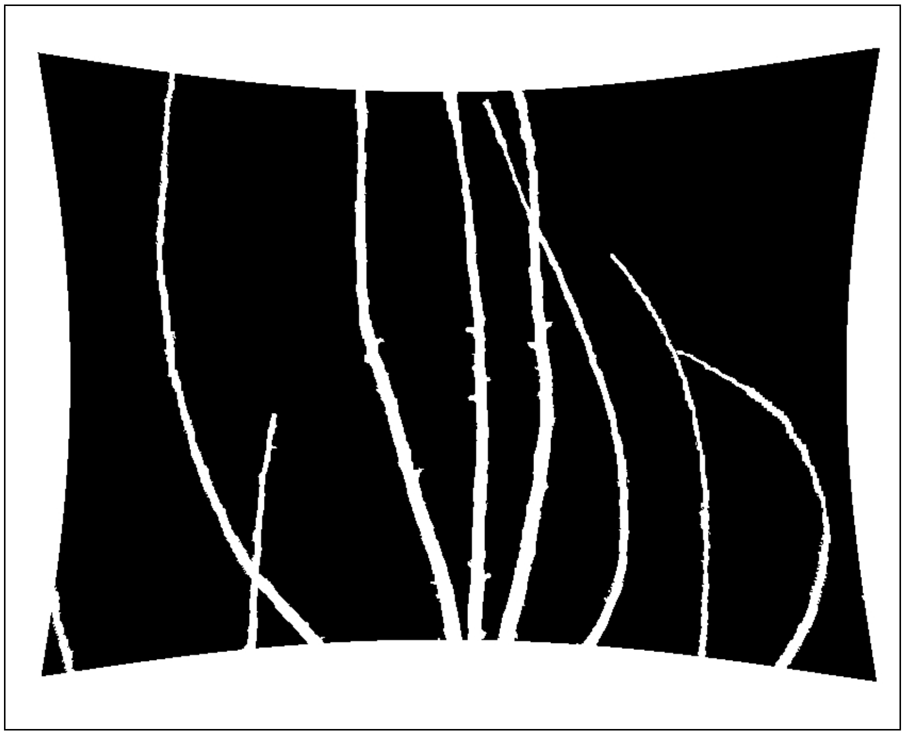
Next, we extract the edges[2] of the undistorted images as sequences of line segments represented as vertices and edges. Then, we convert one edge of the occluding contour into a planar facet as follows. Given that an edge has endpoints $e1$ and $e2$ in world space, we backproject both endpoints a certain distance in front of the image plane to get points p_{e1} and p_{e2}. From these four points, we generate a planar facet represented by vertices $e1$, $e2$, p_{e2}, and p_{e1} and surface normal $\vec{n}$. The planar facet's surface normal $\vec{n}$, when projected to the image plane, points away from the white regions in the image, which represent the object. We create planar facets for all contour edges in the image; this is the viewing cone for each image.

The planar facet generation process is shown in Fig. 9 when the occluding contour is rectangular. Another example, of the temple dataset, is shown in Fig. 10. Fig. 11 shows the viewing cones for five cameras of the temple dataset, where each camera's viewing cone has a unique color. For all of the examples thus far, the total visibility assumption is used. Fig. 12 shows a viewing cone when the total visibility assumption is not in effect.

[2] We mean edges to be the boundary between white and black regions in the image.

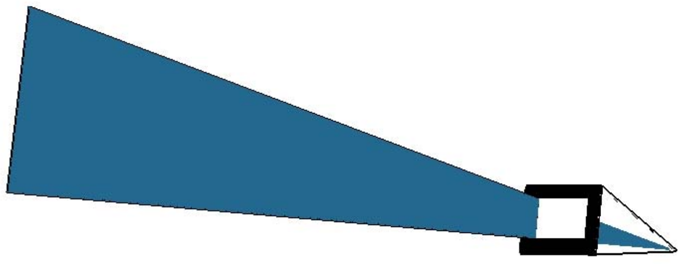


(a) The distorted image is offset by off_x and off_y; the border is filled with white.

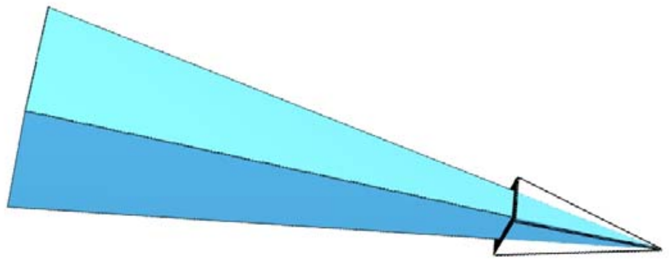


(b) The undistorted image..

Figure 8. Demonstration of the undistortion process.



(a) The planar facet for one edge of the occluding contour.



(b) The planar facets for all edges of the occluding contour.

Figure 9. Construction of planar facets for a rectangular silhouette. The darker blue (subfigure a) represents the back side of the facet, the brighter blue (subfigure b) represents the side of the facet represents the front side, where the surface normal points away from the facet.

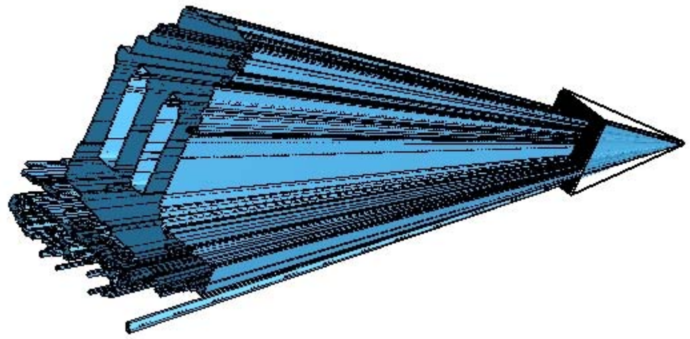


Figure 10. The viewing cone for one camera in the temple dataset; the total visibility assumption is in effect.

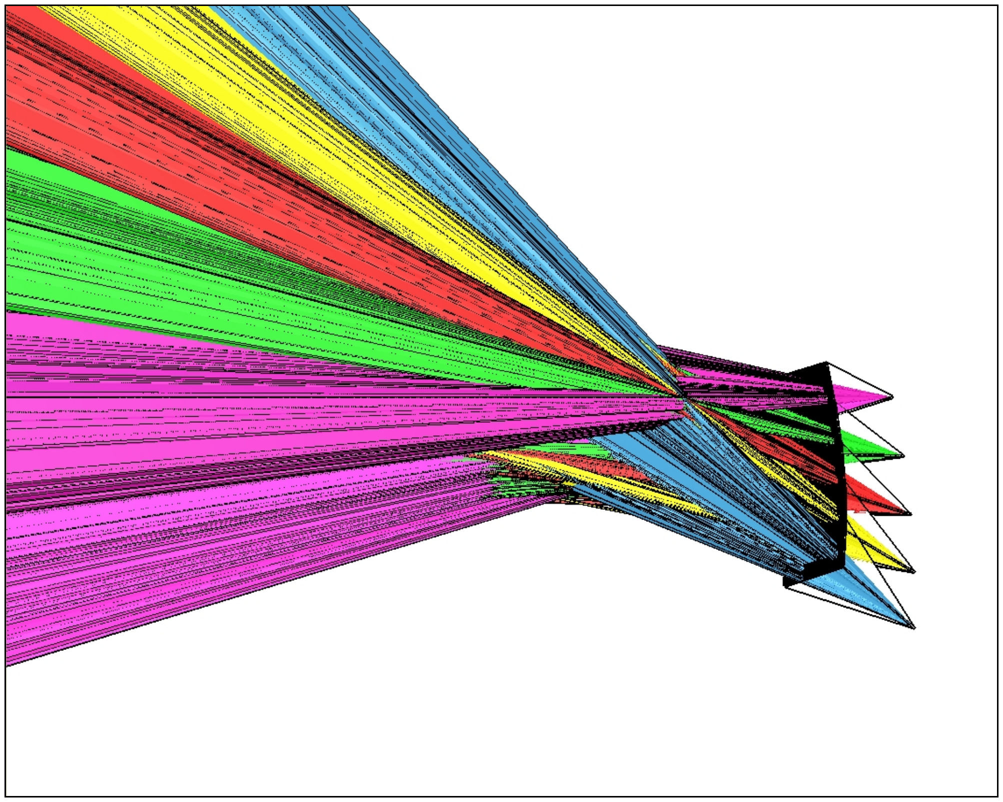


Figure 11. Viewing cones of five cameras in the temple dataset, each viewing cone is a different color. The total visibility assumption is in effect.

4.4 Intersection of viewing cones

Now that the viewing cones for each image have been constructed, it remains to intersect the viewing cones. We have created an algorithm for the intersection of two objects represented by planar facets. We do not go into the details of the algorithm in this article, but plan to discuss the algorithm in a future publication. However, we will show how the algorithm for the intersection of two objects is employed to intersect all of the viewing cones.

We employ a divide and conquer scheme to intersect all of the viewing cones. Let there be N images, and for simplicity in this explanation, assume that N is divisible by two. Then we divide the set of images in half; each set has $N/2$ images in it. We keep dividing until we have sets that contain two viewing cones. Then we intersect the viewing cones in each of the sets. The result is $N/2$ three-dimensional objects, where each object is the intersection of two viewing cones; in other words, each object is the visual hull computed on the basis of only two cameras. Then we divide the set of $N/2$ objects until each set contains two objects. We intersect the two objects, and continue this process until only one object remains, which represents the visual hull for all N images. An example of a typical result when the total visibility assumption is in effect is shown in Fig. 13; the viewing cones that were intersected are shown in Fig. 11. Since the distance between cameras in those figures is small, the reconstruction is not very accurate. Compare those results with a more evenly distributed set of cameras in Fig. 14.

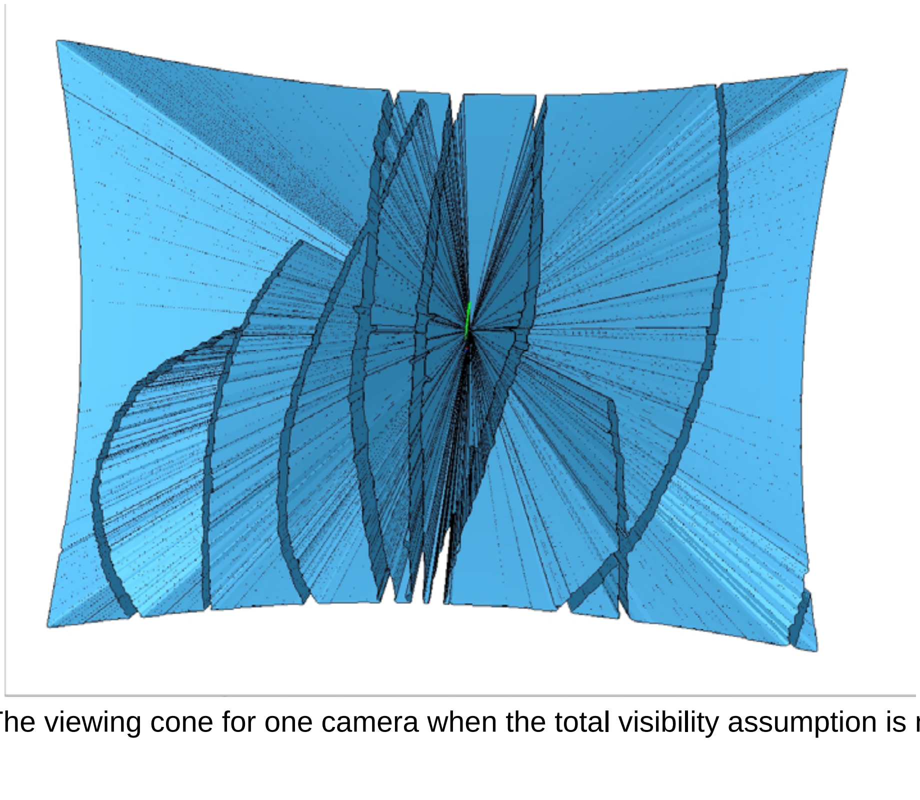


Figure 12. The viewing cone for one camera when the total visibility assumption is not in effect.

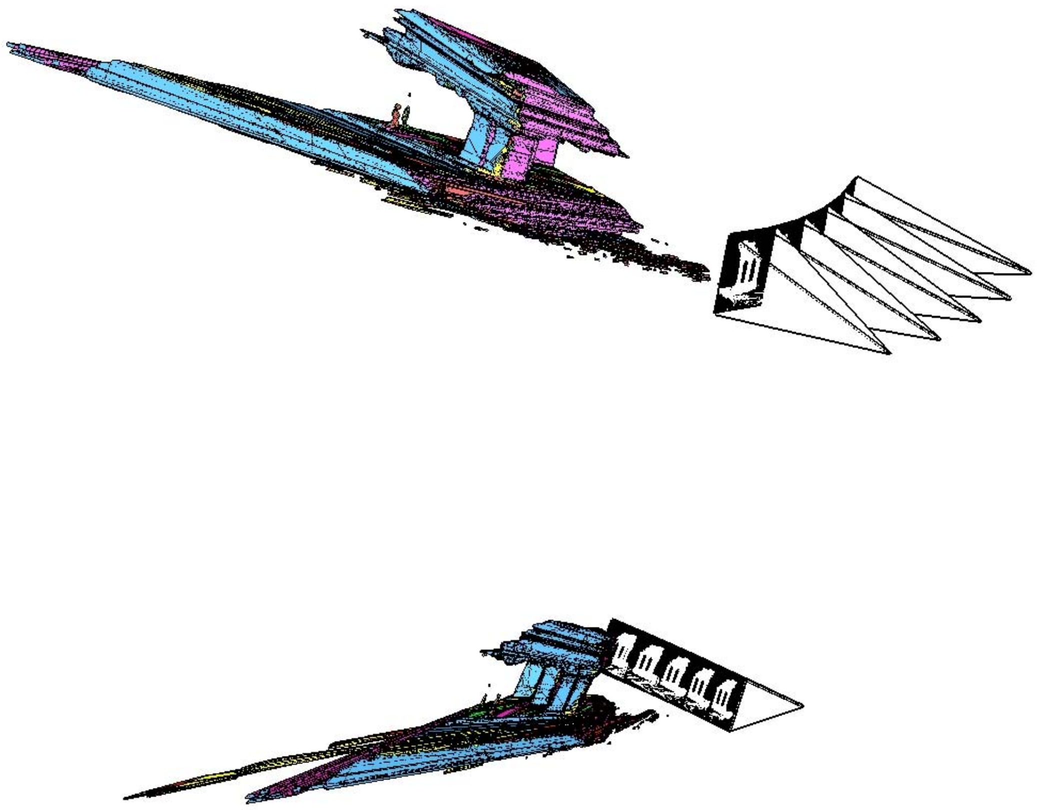


Figure 13. Visual hull computed by intersecting the viewing cones from Fig. 11; two views of the same scene.

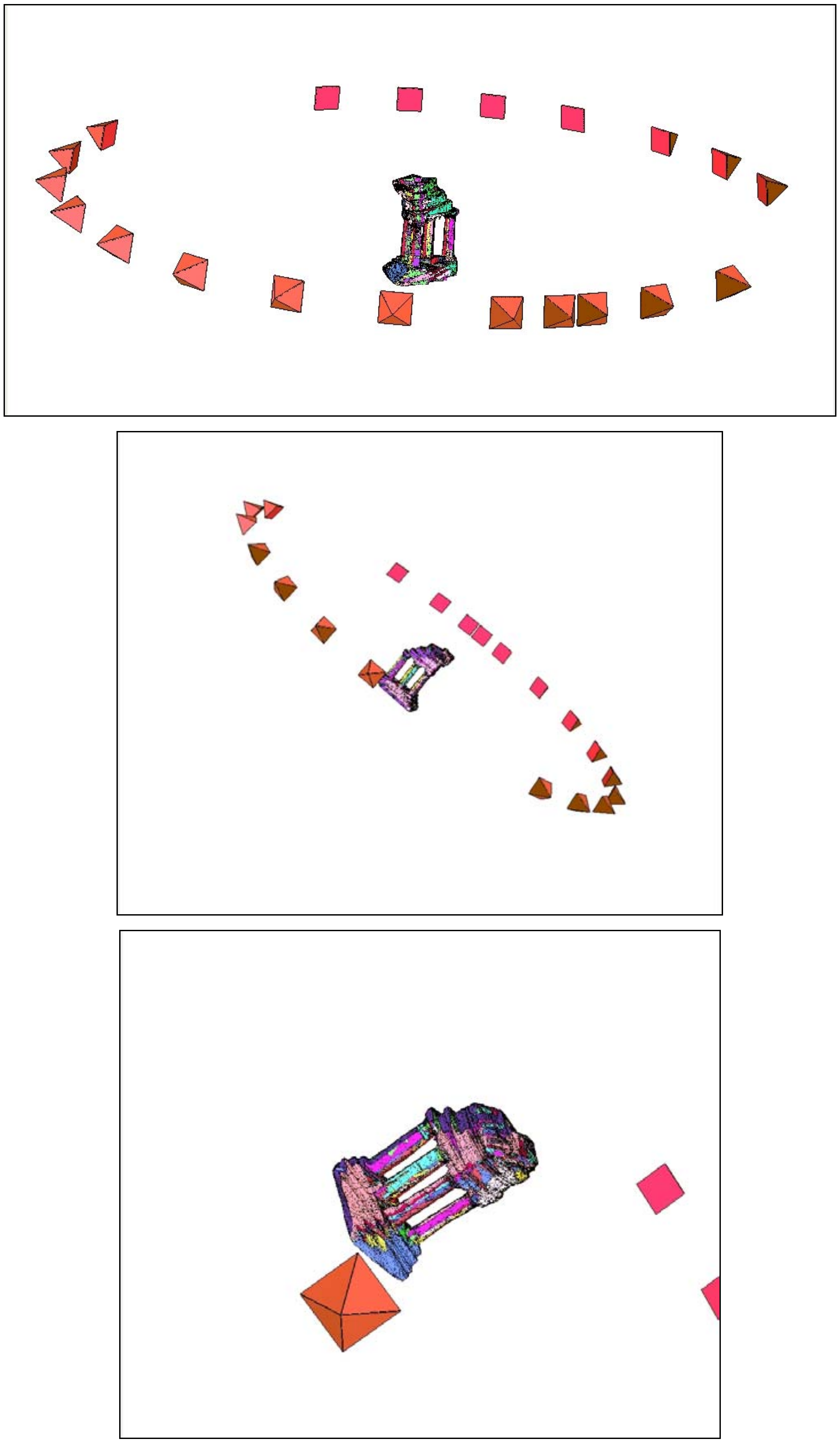


Figure 14. The visual hull of the temple figurine with 20 exactly calibrated cameras; two views and detail. The orange pyramids represent cameras.

5 Experimental Results

In this section we present some results of our SFS method as applied to leafless apple trees. We use a robot (Kawasaki UX-120, Kawasaki Robotics, Wixam, Michigan) and three color cameras (DragonFly, Point Grey Research, British Columbia, Canada). See Fig. 15 for an illustration of the experimental setup. In order to get camera calibration information, we at first relied on the hand-eye calibration of the robot. However, we found that the robot's hand-eye calibration was accurate for very small displacements, but not for larger displacements. Consequently, we gained the camera calibration by moving the robot to the same nine positions in two different passes. We will explain the process in detail here. During the first pass, the tree is not in the scene. The robot is moved to nine known positions; the cameras are calibrated at those nine positions by the typical chessboard calibration pattern. Then, the tree is inserted into the scene and the nine positions are visited again for a second pass; we use the camera calibration information from the first pass as the camera calibration for the second pass. Even though the robot is calibrated, there is enough error in the robot encoder that the calibration of the cameras has some slight error. As has become obvious in the course of this paper, camera calibration error degrades the accuracy of the visual hull. For segmentation, we manually segment the images at this time.

In Fig. 16, we show the results of our method on the leafless apple tree. Notice that it appears that there is a wall surrounding the object; the walls represent the boundary between regions that are seen by at least one camera and the region which no camera can see, and are a consequence of not using the total visibility assumption. The use of more images and wide baselines between cameras would eventually remove the extraneous portions, or what we have been calling walls. In general, the results are encouraging in the sense that the visual hull closely resembles the original object.

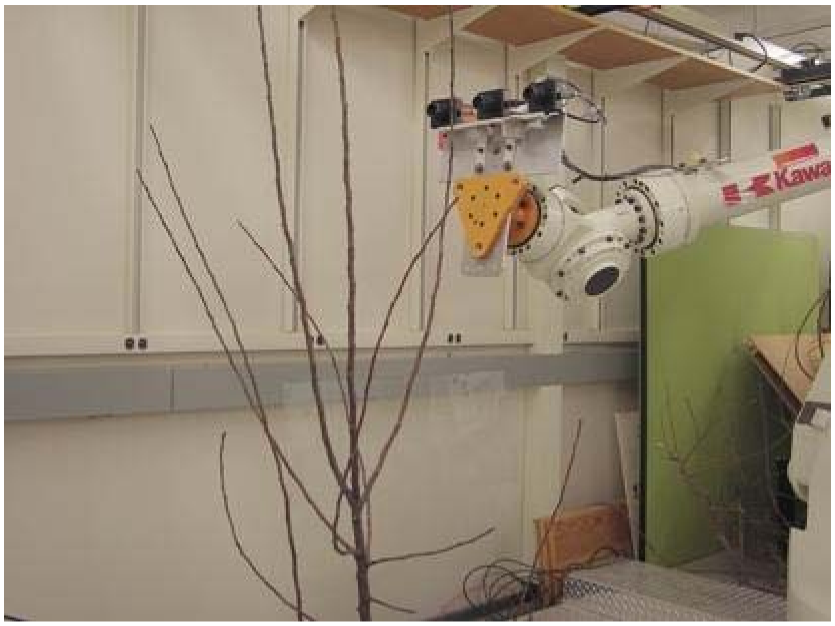


Figure 15. An illustration the leafless apple tree in the workspace and the robot with three cameras.

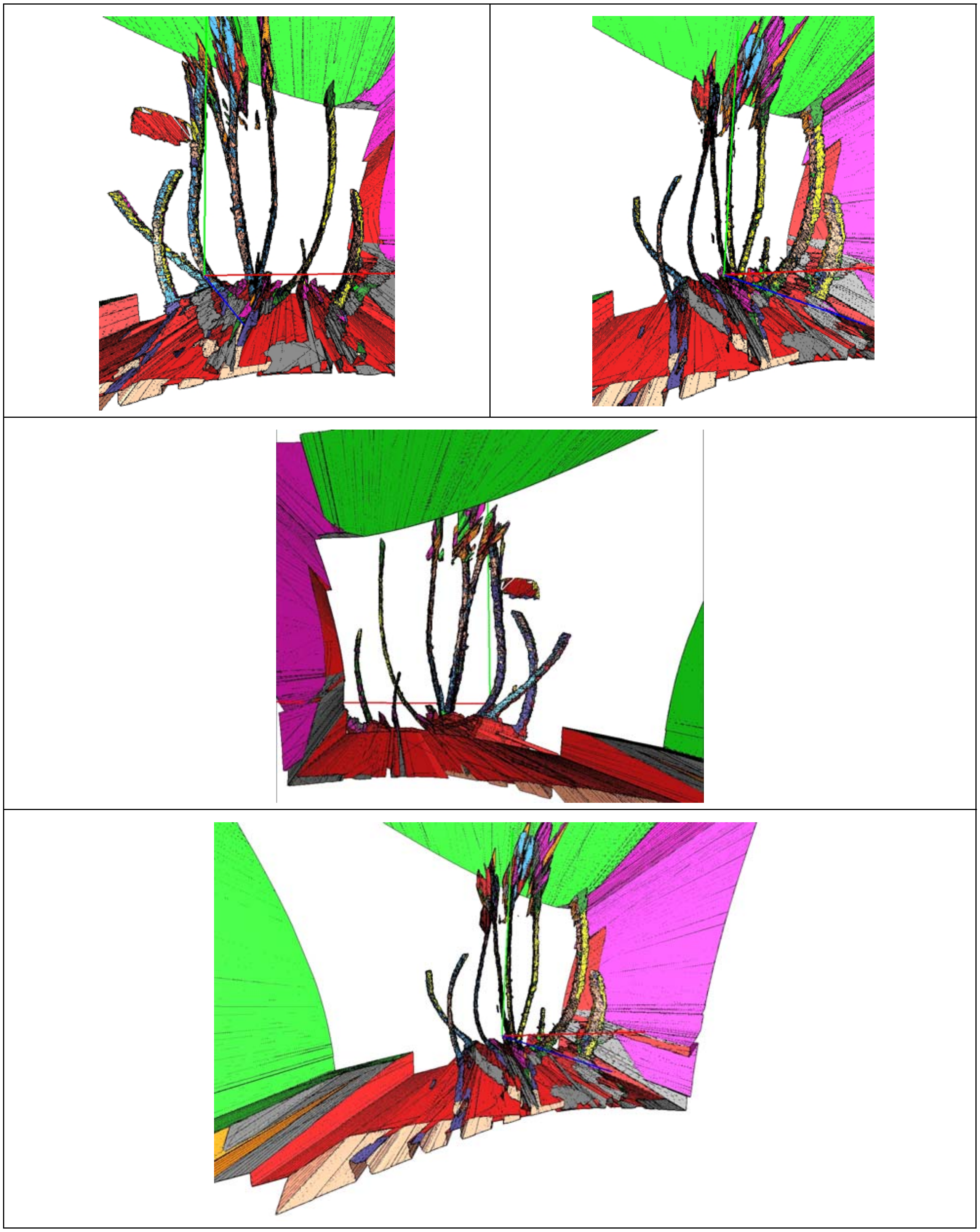


Figure 16. The visual hull of the tree object given 10 cameras that have some calibration error.

6 Discussion and Conclusions

In this paper, we have shown that SFS is a feasible technique for the three-dimensional reconstruction of complex agricultural objects, such as leafless apple trees. Our work for the future is to design an experimental setup that has exact camera calibration, unlike our robotic data acquisition setup. In addition, we wish to explore methods by which the segmentation process can be automated, with the knowledge that an automated segmentation process will have some error. Consequently, our future work will also explore means by which erroneous silhouettes can be corrected or discarded from the visual hull computation. Furthermore, we wish to improve the speed of the algorithm.

Acknowledgements

I would like to thank Prof. Avinash Kak, Prof. Johnny Park, and Purdue's Robot Vision Lab for their comments on this work over the course of several seminars. In addition, I would like to thank Germán Andrés Holguín Londoño for operating the robot and acquiring images for this project.

References

Baumgart, B. G. 1974. Geometric modeling for computer vision. PhD thesis, Stanford University, Stanford, CA, USA.

Boyer, E. and Berger, M.-O. 1997.3d surface reconstruction using occluding contours. Int. J. Comput. Vision, 22(3):219–233.

Boyer, E. and Franco, J.-S. 2003. A hybrid approach for computing visual hulls of complex objects. Computer Vision and Pattern Recognition, Proceedings, 2003 IEEE Computer Society Conference on, 1:I–695–I–701 vol.1.

Chien, C. and Aggarwal, J. 1986. Volume/surface octrees for the representation of three-dimensional objects. Computer Vision Graphics and Image Processing, 36:100–113.

Cipolla, R. and Blake, A. 1990. The dynamic analysis of apparent contours. 1990. Computer Vision, Proceedings, Third International Conference on, pages 616–623.

Cipolla, R. and Blake, A. 1992. Surface shape from the deformation of apparent contours. Int. J. Comput. Vision, 9(2):83–112.

Dyer, C. 2001. Volumetric scene reconstruction from multiple views. In Davis, L., editor, Foundation of Image Understanding, pages 81–100, Boston, MA, USA. Kluwer.

Franco, J.-S. 2005. Three-dimensional modeling from silhouettes. PhD thesis, Institut National Polytechnique de Grenoble.

Franco, J.-S. and Boyer, E. 2003. Exact polyhedral visual hulls. In British Machine Vision Conference (BMVC03), pages 329–338.

Franco, J.-S. and Boyer, E. 2009. Efficient polyhedral modeling from silhouettes. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31(3):414–427.

Giblin, P. J. and Weiss, R. S. 1995. Epipolar curves on surfaces. Image and Vision Computing, 13(1):33 – 44.

Hartley, R. I. and Zisserman, A. 2004. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition.

Koenderink, J. J. 1984. What does the occluding contour tell us about solid shape? Perception, 13(3):321–330.

Koenderink, J. J. and Doorn, A. J. 1976. The singularities of the visual mapping. Biological Cybernetics, 24(1):51–59.

Laurentini, A. 1994. The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell., 16(2):150–162.

Lazebnik, S., Boyer, E., and Ponce, J. 2001. On computing exact visual hulls of solids bounded by smooth surfaces. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 1:I–156–I–161 vol.1.

Lazebnik, S., Furukawa, Y., and Ponce, J. 2007. Projective visual hulls. Int. J. Comput. Vision, 74(2):137–165.

Lazebnik, S. and Ponce, J. 2005. The local projective shape of smooth surfaces and their outlines. Int. J. Comput. Vision, 63(1):65–83.

Liu, X., Yao, H., and Gao, W. 2007. Shape from silhouette outlines using an adaptive dandelion model. Comput. Vis. Image Underst., 105(2):121–130.

Lowe, D. G. 1999. Object recognition from local scale-invariant features. In ICCV ’99: Proceedings of the International Conference on Computer Vision-Volume 2, page 1150, Washington, DC, USA. IEEE Computer Society.

Martin, W. and Aggarwal, J. 1983. Volumetric description of objects from multiple views. Pattern and Machine Intelligence, IEEE Transactions of, 5(3):150–158.

Matusik, W., Buehler, C., McMillan, L., and Gortler, S. J. 2001. An efficient visual hull computation algorithm. Tech. Report, Massachusets Institue of Technology.

Matusik, W., Buehler, C., Raskar, R., Gortler, S. J., and McMillan,L. 2000. Image-based visual hulls. In SIGGRAPH ’00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 369–374, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Potmesil, M. 1987. Generating octree models of 3d objects from their silhouettes in a sequence of images. Comput. Vision Graph. Image Process., 40(1):1–29.

Richards, W. A., Koenderink, J. J., and Hoffman, D. D. 1987. Inferring three-dimensional shapes from two-dimensional silhouettes. J. Opt. Soc. Am. A, 4(7):1168–1175.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 1:519–528.

Slabaugh, G., Culbertson, W., Malzbender, T., and Schafer, R. 2001. A survey of volumetric scene reconstruction methods from photographs. In Proc. International Workshop on Volume Graphics, pages 81–100.

Szeliski, R. 1993. Rapid octree construction from image sequences. CVGIP: Image Underst., 58(1):23–32.

Szeliski, R. and Weiss, R. 1993. Robust shape recovery from occluding contours using a linear smoother. Computer Vision and Pattern Recognition, 1993. Proceedings CVPR ’93., 1993 IEEE Computer Society Conference on, pages 666–667.

Szeliski, R. and Weiss, R. (1998). Robust shape recovery from occluding contours using a linear smoother. Int. J. Comput. Vision, 28(1):27–44.

Vaillant, R. and Faugeras, O. 1992. Using extremal boundaries for 3-d object modeling. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 14(2):157–173.